AMENDMENTS TO THE SPECIFICATION

Please amend the following paragraphs of the Specification as follows.

[0001] The invention relates to computer systems, and more particularly to a method and mechanism for reducing disk input / output operations (IOs) of a computing system by coalescing writes.

[0012] Fig. 1A shows a buffer cache B of a database system A to illustrate an embodiment of the present invention. Each box represents a data block. The Arabic numerals inside the boxes represent the addresses of the data block. An asterisk inside the box indicates the data block is designated dirty. The letters adjacent to and pointing to the boxes represent reference characters.

[0014] Figs. 1C-1I show the temporary buffer C gradually filled with data blocks of adjacent data block addresses according to the embodiment of the present invention. ~~Fig. 1J shows the buffer C, the counter G and a disk E after the data blocks of adjacent data block addresses are written into the disk E.~~

[0014.1] Fig. 1J shows the buffer C, the counter G and a disk E after the data blocks of adjacent data block addresses are written into the disk E.

[0022] According to the present embodiment, whenever a write command is issued by the database system A to write a data block F with a data block address 100 ~~F~~ in the buffer cache B to a disk E, the database system A will search the buffer cache B for additional dirty data blocks that have addresses adjacent to the data block F with the address 100 ~~F~~. In this embodiment, the search space is the entire buffer cache. Various approaches can be taken to search the buffer cache, e.g., using a hashing approach.

[0023] In one embodiment, the search is conducted alternatively on ~~at~~ the lower data block address side and the higher data block address side until the search reaches a not dirty data block respectively at the lower and higher data block address sides or until it reaches the predetermined upper limit of the number of the coalesced data blocks, whichever comes first. In other words, when a data block F with the data block address 100 ~~F~~ is to be written to the disk E, a set of data blocks in the buffer cache B that would form a contiguous space on the disk E with a range that contains the address 100 ~~F~~ is identified. These adjacent data block ~~addresses~~ include, for example, F – n, F – (n-1), . . ., F – 2, F – 1 for the lower data block address side, and F + 1, F + 2, . . ., F + m for the higher data block address side.

[0026] In accordance with an embodiment, when a write command is issued to write the data block F in the buffer cache B to the disk E, the database system A first copies the data block F to a temporary location, such as a buffer C, of the database system A. A count of a count limit counter G is also increased from zero to one. Thereafter, the database system A searches for adjacent dirty data blocks to be written to the disk E. In an embodiment, the database system begins its search by looking for a data block F – 1 at ~~with~~ a next lower data block address 99 ~~F – 1~~. Then the search continues to look for a data block F + 1 at ~~with~~ a next higher data block address 101 ~~F + 1~~. Thus, the data block search according to the embodiment is conducted alternatively on ~~at~~ opposite sides of the lower data block addresses and the higher data block addresses. In other embodiments, the database system A can first search the higher data block address side before it searches the lower data block address side, or it can search for all the adjacent dirty data blocks at one of the data block address sides (e.g., data blocks F – 1, F – 2, F – 3, . . ., etc.) before it moves on to search for adjacent dirty data blocks on the other ~~at another~~ data block address side (e.g., data blocks F + 1, F + 2, F + 3, . . ., etc.).

[0027] When the next lower adjacent data block F – 1 is found and determined dirty, that next adjacent lower data block is copied to the temporary buffer C and the count of the count limit counter G is increased by one (i.e., from 1 to 2). Thereafter, the database system A continues it search to find another adjacent data block in the buffer cache B ~~at~~ on the opposite data block address side. If another adjacent data block, e.g., the data block F + 1, is found and determined dirty, that another adjacent dirty data block is copied to the temporary buffer C and the count of the count limit counter G is again increased by one. After that, the database system A keeps searching alternatively for additional adjacent dirty data blocks in the buffer cache B ~~at~~ on opposite data block address sides. When the database system A can not find a next adjacent data block or the next adjacent data block is determined not dirty, the database system A stops searching for any new data blocks in that direction. For example, if the data block F + 3 is not found or is determined not dirty, the data block F + 3 will not be copied to the temporary buffer C and the database system A will not search further for any additional data blocks at ~~with~~ higher data block addresses (i.e., the data blocks F + 4, F + 5, . . ., etc.). The search will, however, continue in the opposite direction until no adjacent dirty data block is found, or until the count of the count limit counter G reaches the predetermined upper limit.

[0028] An example illustrating an embodiment of the present invention is shown in Figs. 1A-J. As shown in Figs. 1A-B, the database system A comprises a buffer cache B with a plurality of data blocks waiting for being written into the disk E, a buffer C, and a count limit counter G. Some of the data blocks in the buffer cache B have an asterisk "*" ~~a letter "D"~~ included in data block boxes therein, respectively. This asterisk "*" ~~letter "D"~~ indicates that these data blocks are designated as dirty data blocks. Initially, the count of the count limit counter G is reset to zero. When the write command is issued to write the data block F ~~100~~ (i.e., the data block with a data block address 100) into the disk E, the database system A copies the data block F ~~100~~ to the buffer C and the count of the count limit counter G is increased to 1, as shown in Fig. 1C. Thereafter, the database system A continues to alternatively search the buffer cache B ~~at~~ on the lower and higher data block address sides for additional adjacent dirty data blocks until no more adjacent dirty data block is found at both data block address sides or until the count of the count limit counter G reaches the predetermined upper limit, whichever occurs ~~happens~~ first. Fig. 1D shows that a first lower dirty data block F – 1, i.e., ~~a~~ at data block address 99, is found and copied into the buffer C and the value ~~count~~ of the count limit counter G is increased to 2, and Fig. 1E shows that a first higher dirty data block F + 1, i.e., ~~a~~ at data block address 101, is found and copied into the buffer C and the value ~~count~~ of the count limit counter G is increased to 3. Similarly, Figs. 1F-H show that two more lower dirty data blocks (i.e., data blocks F-2 and F – 3 at addresses 98~~,~~ and 97 respectively) and one more higher data block F + 2 (i.e., a data block F + 2 at address 102) are successively found and copied into the buffer C and the count of the count limit counter G is correspondingly increased from 3 to 6. Thereafter, the database system A searches for a data block F + 3 at address 103. Since the data block F + 3 ~~103~~ is not dirty, the data block F + 3 at address 103 will not be copied into the buffer C and the database system A will stop searching for additional data blocks ~~at~~ on the higher data block address side (i.e., no more search for data blocks at upper addresses 104, 105, . . ., etc., even though some of these data blocks exist and are dirty). The database system A, however, continues its search ~~at~~ on the lower data block address side. Fig. 1I shows that the data block F – 4 at address 96 is found and determined dirty and is copied into the buffer C. The value ~~count~~ of the count limit counter G is also increased to 7. If the predetermined upper limit is set to 7, the database system A will then stop searching for any additional data blocks in the buffer cache B. As such, although the buffer cache B still has an adjacent dirty data block F – 5 at address 95, that dirty data block F – 5 at address 95 will not be copied to the buffer C and thus will not be written to the disk E together with those previously identified dirty data blocks at addresses 96-102. Fig. 1J shows the buffer C, the counter G and a disk E after the data blocks of adjacent data block addresses are written into the disk E. Note that after the data blocks are written to the disk, the counter G is reset and contains the value of zero, and the data originally contained in buffer C are flushed.

[0030] According to one embodiment, the present invention adopts a "write cloning" technique to further increase the efficiency ~~efficiently~~ of resource usage in the system. With the write cloning technique, when a data block is identified to be written to disk, a copy of that data block is created at another location in memory. One copy of the data block (e.g., the original copy) is locked and is written to disk. However, the other copy is made available to be accessed by other entities in the system, even during the write operation for its other copy. As a result, these copied data blocks (buffers) are free for future updates by processes of the database system even though the data originally from these data blocks may not yet be actually written to the disk E. And writes will accordingly not be unduly blocking other operations of the database system. Efficiency of the database system is therefore further improved by adopting this "write cloning" technique. It is noted that the "write-cloning" technique described herein can be applied to any computer or computing systems, including any database systems.

[0032] Fig. 2 illustrates a main flowchart of a process for coalescing writes according to one embodiment of the present invention. Box 200 represents that a write command is issued by the database system A to write data in a data block (e.g., the data block F at address 100) to the disk E. At 202, the database system A searches the buffer cache B to identify the data block F at address 100 and confirms that the data block F ~~100~~ is dirty and ready to be written to the disk E. Thereafter, at 204, the data block F ~~100~~ is copied to the buffer C of the database system A, as shown in Fig. 1C. The database system A then marks the data block F ~~100~~ in the buffer cache B as not dirty after it finishes copying the data block F ~~100~~ to the buffer C. After it is marked not dirty, the data block location of the data block F ~~100~~ in the buffer cache B becomes unlock and ready for an update by the database system. Namely, the data block F ~~100~~ in the buffer cache B needs not wait until its data is actually written into the disk E before it can be updated by a process of the database system A. As such, the performance of the database system A is improved and the database system's resources, e.g., the buffer cache B, can be more efficiently used.

[0034] After data of the data block F at address 100 is copied to the temporary buffer C, the database system A searches the buffer cache B for additional dirty data blocks with data block addresses adjacent to the data block F~~100~~, as shown in box 208 of Fig. 2. Subsequently, at 210, the database system A writes all data blocks in the buffer C into the disk E with a single write after all the adjacent dirty data blocks are found or after the count of the count limit counter G reaches the predetermined upper limiter. Thus, the present invention needs just one write IO to write all these identified adjacent dirty data blocks instead of N (N = n + m + 1) individual write IOs for writing each of the data blocks in the buffer cache B. Furthermore, also at 210, the count of the count limit counter G is reset to zero after the data blocks in the buffer C are written into the disk E.

[0035] Fig. 3 shows a flowchart of a process that illustrates detailed actions in performing the box 208 of Fig. 2, according to one embodiment of the present invention. Specifically, Fig. 3 shows detailed actions of box 208 that identifies adjacent dirty data blocks to be written to the disk E. At 300, the database system A starts its ~~it~~ search by looking for a next adjacent dirty data block on ~~at~~ the lower data block address side, i.e., the data block with the data block ~~address~~ F – 1. If the next dirty data block is found at 302, the process goes to box 304. Otherwise, a first flag is set to signal that no more lower adjacent data block is to be identified and the process goes to box 310. At 304, the value ~~count~~ of the count limit counter G is increased by one. The database system A then checks, at box 306, whether the value ~~count~~ is over the predetermined upper limit ~~or not~~. If the count is over the predetermined upper limit, the process continues to box 210. If not, the process goes to box 308. At 308, the currently identified next dirty data block (e.g., the data block F – 1) is copied to the buffer C. And the database system A marks the data block F – 1 in the buffer cache B as not dirty after it finishes copying the data block F – 1 to the buffer C.